PCT/US05/08291

SYSTEM AND METHOD FOR PROVIDING OBJECT TRIGGERS

PRIORITY CLAIM

[0001] The present application claims priority to U.S. Provisional Application No. 60/552,653 filed March 13, 2004, the contents of which are incorporated herein by reference.

RELATED APPLICATIONS

[0002] The present application is related to Attorney Docket Numbers 010-0011, 010-0011A, 010-0011B, 010-0011C, 010-0013, 010-0019, 010-0028 and 010-0030 filed on the same day as the present application. The content of each of these cases is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0003] The present invention relates to triggers in the context of compute resource management and more specifically to a system and method of generating triggers which could be attached to any other scheduling object.

2. Introduction

[0004] The present invention applies to computer clusters and computer grids. A computer cluster may be defined as a parallel computer that is constructed of commodity components and runs commodity software. FIG. 1 illustrates in a general way an example relationship between clusters and grids. A cluster 110 is made up of a plurality of nodes 108A, 108B, 108C , each containing computer processors, memory that is shared by processors in the node and other peripheral devices such as storage discs connected by a network. A resource manager 106A for the node 110 manages jobs submitted by users to be processed by the cluster. Other resource managers 106B, 106C are also illustrated that may manage other clusters (not shown). An example job would be a weather forecast analysis that is compute intensive that needs to have scheduled a cluster of computers to process the job in time for the evening news report.

[0005] A cluster scheduler 104A may receive job submissions and identify using information from the resource managers 106A, 106B, 106C which cluster has available resources. The job would then be submitted to that resource manager for processing. Other cluster schedulers 104B and 104C are shown by way of illustration. A grid scheduler 102 may also receive job submissions and identify based on information from a plurality of cluster schedulers 104A, 104B, 104C which clusters may have available resources and then submit the job accordingly.

[0006] Grid/cluster resource management generally describes the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring grid resources over time in order to run grid applications as efficiently as possible. Each project will utilize a different set of resources and thus is typically unique. In addition to the challenge of allocating resources for a particular job, grid administrators also have difficulty obtaining a clear understanding of the resources available, the current status of the grid and available resources, and real-time competing needs of various users.

[0007] Several books provide background information on how to organize and create a cluster or a grid and related technologies. See, e.g., Grid Resource Management, State of the Art and Future Trends, Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, Kluwer Academic Publishers, 2004; and Beowulf Cluster Computing with Linux, edited by William Gropp, Ewing Lusk, and Thomas Sterling, Massachusetts Institute of Technology, 2003.

[0008] Virtually all clusters have been static which means that an administrator establishes the policies for the cluster, sets up the configuration, determines which nodes have which applications, how much memory should be associated with each node, which operating system will run on a node, etc. The cluster will stay in the state determined by the administrator for a period of months until the administrator takes the entire machine off-line to make changes or modifications. Then the machine is brought back on-line where another 10,000 - 100,000 jobs may be run on it.

[0009] Within this static cluster environment, there is the ability to have something called a job step, a job step allows an application to prepare or modify its environment within the constraints of the compute resources provided by the cluster. For example a job may consist of three steps, the first step is pulling data off of a storage system and transferring the data onto a local file system. The second step may actually process the data and a third step may take the data and go through a second processing step and push it back out to a storage system. These job steps enable some additional functionality for the job in that it allows a job to work within the environment they have.

[0010] However, there are some deficiencies in this process. Using job steps does nothing for allowing the jobs to actually change the compute environment provided by the cluster in any way. Job steps operate within the cluster environment but have no control or ability to maximize efficiencies within the environment or adjust the environment. They are static in the sense that they are limited to manipulation of tasks within the given cluster environment. What is needed in the art is a method of improving the efficiency of the compute environment via a device associated with a job or other object.

SUMMARY OF THE INVENTION

[0011] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth herein.

[0012] The present invention addresses the deficiencies in the art discussed above. The cluster that receives a job submission according to the present invention is dynamic in that the cluster and the resources associated with the cluster may dynamically modify themselves to meet the needs of the current workload. To accomplish this dynamic component of the cluster, the present invention further involves introducing triggers.

[0013] A trigger is an object which can be attached or associated with any other scheduling object. A scheduling object can be, for example, one of: a compute node, compute resources, a reservation, a cluster, user credentials, groups or accounts, a job, a resource manager, other peer services and the like. Any scheduling object can have any number of triggers associated with it.

[0014] The invention comprises various embodiments associated with dynamic clusters and triggers. These embodiments include systems, methods and computer-readable media that provide the features of the invention. The method embodiment of the invention comprises a method for dynamically modifying a cluster, the method comprising attaching a trigger to a scheduling object and firing the trigger based on a trigger attribute, wherein the cluster environment is modified by an action take by the trigger.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0016] FIG. 1 illustrates generally a prior art arrangement of clusters in a grid;

[0017] FIG. 2 illustrates a trigger attached to an object;

[0018] FIG. 3 illustrates an example of the user of triggers according to an aspect of the invention;

[0019] FIG. 4 illustrates a method according to an embodiment of the invention; and

[0020] FIG. 5 illustrates a graphical user interface used to create triggers.

DETAILED DESCRIPTION OF THE INVENTION

[0021] Various embodiments of the invention are discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

[0022] The "system" embodiment of the invention may comprise a computing device that includes the necessary hardware and software components to enable a workload manager or a software module performing the steps of the invention. Such a computing device may include such known hardware elements as one or more central processors, random access memory (RAM), read-only memory (ROM), storage devices such as hard disks, communication means such as a modem or a card to enable networking with other computing devices, a bus that provides data transmission between various hardware components, a keyboard, a display, an operating system and so forth. There is no restriction that the particular system embodiment of the invention has any specific hardware components and any known or future developed hardware configurations are contemplated as within the scope of the invention when the computing device operates as is claimed.

[0023] The present invention enables the dynamic modification of compute resources within a compute environment such as a cluster or a grid by the use of triggers. FIG. 2 illustrates a trigger 204 being attached to an object 202. The object 202 is preferable a scheduling object and each trigger 204 is configured with a plurality of attributes. Example objects include a compute node, a reservation within a cluster, a cluster itself, a user, a job submitted by a user to a cluster manager, a resource manager, etc. As can be appreciated, an "object" in the context of cluster management may be any number of concepts to which a trigger may be attached.

[0024] An example attribute associated with a trigger includes an event type, which means that one would like this trigger to fire or execute based on a particular event occurring such as the creation of the object, the starting, execution, cancellation or termination of an object, or an object state.

[0025] Other attributes associated with a trigger include a time-out, an offset feature, a particular action (such as send an e-mail to the administrator), dependencies, an argument list, a state and a threshold value. This is not meant to be an exhaustive complete list. Other attributes may also be attached to the trigger.

For example, meaning dependencies can be based on attributes within the object, wherein if a job is now running, a dependency may be that it fires if a parameter is set to "true". In that case, the trigger also has a variable it sets to cascade other triggers by setting variables that cause other triggers to fire. Such parameters may relate to things like a threshold, a re-arm time, time-out values and durations. In this manner, a cascade of triggers may fire based on various modified and set parameter from one trigger to the next. Other values that may be used to fire triggers include such parameters as: user credentials, jobs, groups, jobs per user and other types of thresholds. For example, whenever a user exceeds X number of jobs, launch a trigger to take an action. A group-based parameter example is: (1) if user John has more than 18 idle jobs, then send a note to an administrator; and (2) if a group "staff" resource availability query receives a reply with resources more than two hours out, then launch a trigger to modify reservation Y to provide more resources.

[0026] The offset feature involves establishing that the trigger will fire either before or after an event has occurred. The example trigger in FIG. 3 illustrates their use in a hosting environment in which a customer wants to reserve a block of resources for a particular time frame and the administrator wants to dynamically provision those resources. FIG. 3 illustrates a reservation 302 that is processing in time. A trigger 304 is attached to the object with attributes including an offset to begin a certain period of time (say two minutes) 312 after the reservation 302 begins its process. The trigger 304 has as an attribute an action to take which is to set up a network and generate an ARGLIST variable called $IPlist and return that value to the reservation environment. The trigger 304 also transmits the $IPList to another trigger 306. The trigger 306 has a start time offset but also a dependency that it does not fire until the $IPList variable is set. Once the variable is set, the trigger 306 sets up a storage area network, brings in the resources and makes the resources available to the reservation. When trigger 306 completes, a third trigger 308 performs an operating system setup, which also has a dependency on the $IPList variable being set to a value as well as a variable being set to "true". When both of those parameters are satisfied, trigger 308 fires and sets up the operating system and application environment and completes. The output of trigger 308 is a parameter stating whether the operating system setup was successful ("true") or not.

[0027] Independent of these triggers is an additional trigger 310 that is set to fire at a fixed offset from the start of the reservation, and it performs a health check to verify that the OS setup variable which is setup by the trigger 308 is true. If it is not set to true, then trigger 310 is designed to do two things: (1) cancel the reservation itself and send an e-mail to the administrator and end user notifying them that there has been a failure and the reservation will not be available; and (2) retry the initial setup triggers or look for additional local in time at which these blocked resources could be made available and send an e-mail to the user saying we'll retry at this particular time. All of this is performed automatically through the use of triggers.

[0028] The above example provides an illustration of the various features of triggers, including the ability to start at an offset value, perform certain actions, having certain dependencies based on data being processed and received or other kinds of dependencies and produce and receive argument lists.

[0029] In addition, triggers can specify arbitrary actions allowing it to modify the scheduling state, to execute some process, to pull something in from off the Internet or to update a database. Any arbitrary action that can modify the environment, including destroying the object or reconfiguring the object. Furthermore, triggers have the ability to specify dependencies, saying the trigger can only fire when an event has occurred, the offset has been satisfied and certain other conditions such as variables have been set or other triggers completed with certain states. Each trigger can begin with a variable called in from an ARGLIST which allows you to pass in either static or dynamic variables to modify its behavior.

[0030] Also associated with triggers is the concept of a trigger timeout. This feature allows one to determine if a trigger has not fired yet or if it has completed successfully, unsuccessfully or if it's still in process of completing. With all these capabilities, an administrator can have essentially arbitrary control over decision making and process flow to modify the dynamic cluster environment in any way desired.

[0031] There are a number of ways to create a trigger. FIG. 5 illustrates a graphical tool 500 to simply point and click to associate the trigger and attach it to an object. The tool allows the user to select: the creation of a trigger when a reservation starts (or other selectable time via a drop down menu) 502, the trigger start time for a certain number of minutes before or after a reservation starts 504, an action launched by a trigger such as to cancel the reservation 506, an executable file to execute 508 or to receive an argument list 510 and a reservation utilization threshold 512.

[0032] Any action may launch a trigger. For example, if a resource manager goes down, or is a software license is about to expire, or a software application that is going to have a job executed with use of the software and it is out-of-date. Any event may launch a trigger.

[0033] The second method is to set it up in a configuration file a Moab™ configuration file is simply a flat text file which specifies associations and definitions of triggers. A third way is to simply use command line arguments to generate a trigger. These triggers can be created remotely over the network interface or locally. The following is an example of a command line method of creating triggers by user "Smith":

```
mrsvctl -c -h smith     -T \

'Sets=$Var1.$Var2.$Var3.!Net,EType=start,AType=exec,Action=/tmp/Net.sh,Timeout=1
0'\
                -T \
```

```
Requires=$Var1.$Var2.$Var3,Sets=$Var4.$Var5,EType=start,AType=exec,Action=/tmp/
FS.sh'\
                    -T \
Requires=$Var1.$Var2.$Var3.$Var4.$Var5,Sets=!NOOSinit.OSinit,Etype=start,AType=exe
c,Action=
/tmp/OS.sh+$Var1:$Var2:$Var3:$Var4:$Var5' \
                    -T \
Requires=failed,AType=cancel,EType=start \
                    -T \
Eype=start,Requires=OSinit,AType=exec,Action=/tmp/success.sh\
                    -T \
Requires=Net,EType=start,Sets=failed,AType=exec,Action=/tmp/fail.sh
```

[0034] This demonstrates a string of triggers, the first two set variables, the third one requires each of those variables to be set and there are also triggers that activate in case of failure.

[0035] An important feature that differentiates triggers from the job step is that there are other systems that allows one to have some sense of dependencies and modification but that is only within a single, given application or job. Job steps can modify their own data and the like but there's nothing that can modify either scheduling policy or scheduling objects, or scheduling environment, like triggers can. Triggers allow one to take any arbitrary action based on any arbitrary set of sensors. Triggers enable pulling in a wide ranging scope of information and having a wide scope of control. They are preferable written in the "c" programming language but there are no constraints on the type of programming language.

[0036] One of the attributes introduced above that is associated with a trigger is the threshold attribute. In addition to being able to say that a trigger will fire, when its dependencies are satisfied and its event has occurred and its offset has been satisfied, one may also specify whether a particular threshold and its threshold criteria has been satisfied. This feature allows one to have triggers that fire when particular qualities of service are not satisfied, when queue times have been exceeded, when anything that correlates to basically system performance has or has not been satisfied. When these metrics have not been satisfied or have been satisfied this provides some way one can have arbitrary actions occur.

[0037] Other examples of trigger usage are that an administrator can attach a trigger to a node and allow a node monitor such as Ganglia to perform monitoring activities such as detecting keyboard touches. So if a local user has begun to type or if the system detects a high level of data transmission or swapping, a trigger action may adjust the priority of that node so that it is no longer as likely to be selected for batch work load. The priority adjustment may reduce the probability that the node would be selected for a large job like a batch work load.

[0038] Performance triggers illustrate another type of trigger that is associated with a particular group or a particular user and a threshold parameter. The parameter may be a performance threshold parameter that is related to, for example, an average response time that is below a particular threshold. If that particular threshold is not satisfied, then the trigger fires and sends an e-mail off to an administrator and adjusts the priority of that user's jobs. The trigger may also dynamically modify the cluster resources to accommodate the at least one user's activities so that the user experiences a performance level at least at the threshold parameter.

[0039] Embodiments within the scope of the present invention may also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or combination thereof) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of the computer-readable media.

[0040] Computer-executable instructions include, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Computer-executable instructions also include program modules that are executed by computers in stand-alone or network environments. Generally, program modules include routines, programs, objects, components, and data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represents examples of corresponding acts for implementing the functions described in such steps.

[0041] Those of skill in the art will appreciate that other embodiments of the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. Embodiments may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination thereof) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0042] Although the above description may contain specific details, they should not be construed as limiting the claims in any way. Other configurations of the described embodiments of the invention are part of the scope of this invention. Accordingly, the appended claims and their legal equivalents should only define the invention, rather than any specific examples given.